Remarks

The above Amendments, these Remarks, and this request for continued examination is in response to the Office Action mailed July 24, 2007 and the Examiner Interview of October 16, 2007. Applicant acknowledges with thanks Examiner Rampuria's assistance in granting an interview on October 16, 2007, during the course of which interview various features of the claimed embodiments were discussed, the substance of which is included herein.

I. Summary of Examiner's Objections/Rejections

The Office Action rejected claims 8-16 under 35 U.S.C.102(e) as being anticipated by Raventos (U.S. Pub. No. 2002/0194244).

Claims 8-16 were provisionally rejected on the ground of non-statutory obviousness-type double patenting over claims 1-11 of co-pending Application No. 10/788,802.

II. Summary of Applicant's Response

The present Reply amends claims 8 and 13, cancels claims 10-12, and adds new claims 17-30, leaving for the Examiner's present consideration claims 8-9 and 13-30. Reconsideration of the rejections is requested.

III. Brief Summary of Differences Between the Claims and the Prior Art

In order to protect against applications that share a logical connection from making concurrent updates to a resource that cannot tolerate concurrent updates to the resource, the claims define a method for serializing enlistments of resources across a shared logical connection. Figure 3 shows an interleaving enlistment scenario where the enlistment request for thread T2 is blocked until

the in-progress enlistment of thread T1 completes. Without blocking the second thread from calling XAResource.start() on the resource, the second start operation would fail with an exception. The exception would be generated because it is an XA protocol violation to have different transactions enlist the same resource across a shared logical connection at the same time. In order to prevent a second thread from calling XAResource.start() on the resource through the shared logical connection, each XAResource instance is wrapped in an object that the transaction manager will use to synchronize concurrent enlistment requests. The transaction manager maintains a collection of these wrapped objects, which is consulted on each resource enlistment. Each request to enlist the resource will first check to see if there is a lock being held on the resource by another thread of control. If not, the lock is granted to the accessor and held until the owner delists the resource. The waiting threads, if any, are then signaled that the lock is free. One of the waiting threads will be granted the lock and will be allowed to proceed with its enlistment. The collection of wrapped XAResource objects is periodically garbage collected to clear stale and unused entries.

Raventos discloses enabling a transaction-based service utilizing non-transactional resources. While Raventos discusses transaction processing, the XA protocol, and resource managers, Raventos does not discuss the problem of how to synchronize concurrent resource requests across a shared logical connection to an XAResource that can not tolerate concurrent access.

IV. Response to 35 U.S.C. 103(a) Rejections to Claims 8-9 and 13-16

Independent Claim 8

Independent Claim 8 (as amended) states:

A method for protection against interleaving transactions, comprising:

communicating with a resource manager from an application using an Application Programming Interface (API), wherein ~~the API~~ multiple threads of the application utilize ~~utilizes~~ a shared logical connection to the resource manager;

controlling transaction demarcation using the Java™ Transaction API (JTA);

communicating with the resource manager from a first thread of a transaction manager during two phase commit processing using an XAResource interface;

enlisting a resource, wherein the first thread of the transaction manager associates a unique transaction identifier with work that is performed on the [[a]] resource by invoking XAResource.start() on the resource and subsequent application updates to the resource are associated with a global transaction, wherein the resource is wrapped in an object that the transaction manager uses to synchronize concurrent enlistment requests, and wherein the transaction manager maintains a collection of wrapped XAResource objects which is consulted on each resource enlistment;

delisting a resource, wherein the first thread of the transaction manager invokes XAResource.end() on the resource and future application updates on the resource over the shared logical connection are disassociated from the global transaction; and

blocking a second thread of the transaction manager from calling XAResource.start() on the resource until the first thread of the transaction manager has called XAResource.end() on the resource.

Claim 8 requires blocking a second thread of the transaction manager from calling XAResource.start() on the resource until the first thread of the transaction manager has called XAResource.end() on the resource. The office action cited Raventos as disclosing this feature in paragraphs 56-58, but while the cited portion of Raventos mentions XAResource.start() and XAResource.end(), the cited portion does not teach or suggest blocking a second thread of the transaction manager from calling start until the first thread of the transaction manager has called end().

Claim 8 also requires that the transaction manager maintains a collection of wrapped XAResource objects which is consulted on each resource enlistment. The office action cites paragraph 9 of Raventos for disclosing this feature, where it states, "Once the services are enlisted to be part of a transaction, the Resource Manager takes care of interfacing the bus' Transaction Manager in the sequence of events defined by a transaction protocol (e.g., the XA protocol), and to appropriately invoke the proper plugins to execute the service, validate it, undo it, or check for complete rollback of the service." While the cited portion discusses a resource manager for an XA resource, the cited portion does not discuss a wrapped XAResource object, nor does it discuss a collection of wrapped XAResource objects being maintained to be consulted on each resource enlistment. Raventos does not teach or suggest maintaining a collection of wrapped XAResource objects which is consulted on each resource enlistment.

Applicant respectfully submits that the embodiment as defined in Independent Claim 8 is neither anticipated by nor obvious in view of Raventos.

Dependent Claims 9 and 13-16

Dependent Claims 9 and 13-16 depend from Claim 8. For at least the reasons discussed above with regards to Claim 1, dependent Claims 9 and 13-16 are also patentable. Dependent claims 9 and 13-16 add their own features which render them patentable in their own right.

Dependent Claim 16

Dependent Claim 16 requires that the collection of wrapped XAResource objects is periodically garbage collected to clear stale and unused entries. The office action erroneously asserted that this feature was disclosed by paragraph 65 of Raventos "if a non-transactional resource is performing a task that may be easily undone (or rolled back), then the Resource Manager may follow the Online Mode of operation for such resource." The cited portion of Raventos teaches using an online mode of operation for non-transactional resources that can be easily rolled back. The cited portion of Raventos does not discuss using garbage collection on a collection of wrapped XAResource objects.

V. Conclusion

In light of the above, it is respectfully submitted that all of the claims now pending in the subject patent application should be allowable, and a Notice of Allowance is requested. The Examiner is respectfully requested to telephone the undersigned if he can assist in any way in expediting issuance of a patent.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 06-1325 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: November 19, 2007

By: /Thomas K. Plunkett/
Thomas K. Plunkett
Reg. No. 57,253

Customer No. 23910
FLIESLER MEYER LLP
650 California Street, 14th Floor
San Francisco, California 94108
Telephone: (415) 362-3800